Technical Report

EasyTCGA: An R package for easy batch downloading of TCGA data from FireBrowse

Viktoria Kliewer & Sangkyun Lee

09/2016

technische universität dortmund

SFB 876 Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung

1 Introduction

Many organizations deal with the investigation of cancer including the National Institutes of Health, USA. Genomics(CCG)[1]. The Cancer Genome Atlas (TCGA)[2] is an establishment of the National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI) that has created maps of the key genomic changes in more than 30 cancer types. The aim of TCGA is the improvement of the effectiveness to diagnose, to treat and to guard against cancer through genome analysis. TCGA provides a publically available dataset.

The Broad Institute TCGA GDAC Firehose[3] arranges this data set that can be loaded directly with use of FireBrowse[4]. FireBrowse allows simple and smart download and study TCGA data and TCGA analyses. The data is downloaded as zip files. Mario Deng created an R client called *FirebrowseR* with the objective of getting the TCGA data from FireBrowse conveniently. The size of record sets to download is limited. *EasyTCGA* is an R package providing easy batch downloading of particular TCGA data from FireBrowse using *FirebrowseR*. The key advantage of *EasyTCGA* is the downloading of the whole available data set you are interested in at once as a single data frame.

The focus of this technical report is on the presentation of the R package *EasyTCGA*. That is why all specific expressions and variables like biological data and the like will not be explained. You get all relevant background informations on the given URL's. *EasyTGCA* can download clinical data, sample-level log2 miRSeq and mRNASeq expression values, selected columns from the MAF (Mutation Annotation File) generated by MutSig and significantly mutated genes, as scored by MutSig.

1.1 Installation

The package is available at GitHub[5]. Type the following command in R console:

```
> install.packages("devtools")
> devtools::install_github("sanglee/EasyTCGA")
```

Before downloading data with *FirebroweR* and *EasyTCGA*, it is recommandable to check if the data to query is available on FireBrowse[4].

2 Functionality of *EasyTCGA*

This section gives an overview of the R package *EasyTCGA* to acquaint the user with this package and to get started to use it. We will present the functions, some sensible

[1] http://www.cancer.gov/about-nci/organization/ccg
[2] http://cancergenome.nih.gov/
[3] https://gdac.broadinstitute.org/
[4] http://firebrowse.org/
[5] https://github.com/sanglee/EasyTCGA

steps and codes and the respective outputs. You find the detailed description in the documentation of the package.

First, load the *EasyTCGA* package:

```
> library(EasyTCGA)
```

All packages required for this package, including how to install *FirebrowseR*, are loaded automatically. Note that entries in the downloaded data frame can be "NA", "None", empty or the output is even "NULL" if the data for your query is incomplete or there is none.

2.1 Clinical data

To deal with clinical data, we have functions in *dn_clinical.R*. Especially these provide data for all available cancer types, called "cohorts", and the TCGA "patient barcodes" of single, multiple or all cancer types. These parameters, namely cohorts and patient barcodes, are the most important input arguments for the functions of *EasyTCGA*.

The command

```
> cohorts = dn_cohorts()
> cohorts
 [1] "ACC"  "BLCA" "BRCA"   "CESC" "CHOL" "COAD"  "COADREAD" "DLBC"
    "ESCA"
[10] "FPPP" "GBM"  "GBMLGG" "HNSC" "KICH" "KIPAN" "KIRC"     "KIRP"
    "LAML"
[19] "LGG"  "LIHC" "LUAD"   "LUSC" "MESO" "OV"    "PAAD"     "PCPG"
    "PRAD"
[28] "READ" "SARC" "SKCM"   "STAD" "STES" "TGCT"  "THCA"     "THYM"
    "UCEC"
[37] "UCS"  "UVM"
```

outputs a character vector containing all TCGA cohort abbreviations which are relevant for the algorithms. Use the function *dn_clinical_one* to download all available clinical data of a single cohort. Specify the input argument `cohort`, e.g. BRCA:

```
> cohort = "BRCA"
> brca.clinical = dn_clinical_one(cohort)
```

At present, the `brca.clinical` is a data frame of 1097 observations (patients) and a11 variables, these are clinical data elements (CDEs) like gender, age, race, duration of the illness and biological data. All available CDEs can be downloaded as a character vector using the function *Metadata.ClinicalNames* of *FirebrowseR*:

```
> clinical.names = Metadata.ClinicalNames(format = "csv")
```

An extract of the data frame `brca.clinical` querring information about the CDEs `tcga_participant_barcode, days_to_birth` and `bcra_canonical_reason`:

```
> brca.clinical[1:3, c("tcga_participant_barcode", "days_to_bi-
rth", "bcr_canonical_reason")]
```

```
   tcga_participant_barcode  days_to_birth  bcr_canonical_reason
1              TCGA-3C-AAAU         -20211                  <NA>
2              TCGA-3C-AALI         -18538                  <NA>
3              TCGA-3C-AALJ         -22848                  <NA>
```

There is also a possibility of downloading clinical data of multiple cohorts or all available clinical data as a single data frame applying the function *dn_clinical*. The command

```
> all.clinical = dn_clinical(cohorts)
```

returns all available clinical data.

To extract TCGA patient barcodes as a character vector from downloaded clinical data, as `brca.clinical` or `all.clinical`, use

```
> brca.barcodes = patient_barcodes(brca.clinical)
> all.barcodes = patient_barcodes(all.clinical)
```

2.2 Sample-level log2 miRSeq and mRNASeq data

This section presents the approach of the files:

- *dn_miRNA.R*, consisting of *dn_{miRSeq, miRSeq_cohort}*
- *dn_mRNA.R*, consisting of *dn_{mRNASeq, mRNASeq_cohort}*

All data frames of sample-level log2 miRSeq data provide information about the variables `tcga_participant_barcode, mir` (a micro ribonucleic acid, miRNA), `expression_log2`, `tool`, `cohort`, `sample_type` and `date`. The variables of sample-level log2 mRNASeq data are `tcga_participant_barcode, gene` (messenger RNA, mRNA), `expression_log2`, `z.score`, `cohort`, `sample_type`, `protocol` and `geneID` (mRNA).

The function *Metadata.SampleTypes* of *FirebrowseR* returns all TCGA sample type codes, both numeric and symbolic.

```
> sample.types = Metadata.SampleTypes(format = "csv")
```

We download data of all sample types available.

There are the R files *miRNA_ID.R* and *mRNA_ID.R* which provide all available mirs and genes. The ID's are as follows:

```
> length(miRNA_ID)
[1] 2588
>  miRNA_ID[1:5]  # provided by the package
[1] "hsa-let-7a-2-3p" "hsa-let-7a-3p" "hsa-let-7a-5p"
[4] "hsa-let-7b-3p"   "hsa-let-7b-5p"
> length(mRNA_ID)
[1] 20501
> mRNA_ID[1:5]  # provided by the package
[1] "A1BG"  "A1CF"  "A2BP1" "A2LD1" "A2M"
```

The function *dn_miRSeq* with input parameters `tcga_participant_barcode`, `mir`, `cohort`, `page.Size` and `sort_by` is intended to download rather small and diverse data sets, e.g. specifying just some mirs, cohorts and barcodes.

```
> mir = miRNA_ID[1:10]
> cohort = "BLCA"
> tcga_participant_barcode = " TCGA-ZF-AA53 "
  # a TCGA patient barcode from BLCA
> page.Size = 250
> sort_by =  "tcga_participant_barcode"
> obj = dn_miRSeq(mir,cohort,tcga_participant_barcode,sort_by,
page.Size)
```

The `obj` is a data frame with 5 observations of 7 variables.

```
> obj
   tcga_participant_barcode            mir  expression_log2
1             TCGA-ZF-AA53   hsa-let-7b-3p        4.2048666
2             TCGA-ZF-AA53   hsa-let-7b-5p       13.4371320
3             TCGA-ZF-AA53 hsa-let-7a-2-3p        0.7719075
4             TCGA-ZF-AA53   hsa-let-7a-3p        4.7987074
5             TCGA-ZF-AA53   hsa-let-7a-5p       14.3184761

                      tool cohort sample_type                  date
1 miRseq_Mature_Preprocess   BLCA          TP   2016-01-28 00:00:00
2 miRseq_Mature_Preprocess   BLCA          TP   2016-01-28 00:00:00
3 miRseq_Mature_Preprocess   BLCA          TP   2016-01-28 00:00:00
4 miRseq_Mature_Preprocess   BLCA          TP   2016-01-28 00:00:00
5 miRseq_Mature_Preprocess   BLCA          TP   2016-01-28 00:00:00
```

In contrast, *dn_miRSeq_cohort* with input parameters `cohort` and `page.Size` supplies all available sample-level log2 miRSeq expression values of one cohort as a single data frame. Remark that these data sets are quite big, so especially for cohorts with a large number of patients the download will take much time. The command

```
> cohort = "TGCT"
> page_size = 2000
> tgct.miRSeq = dn_miRSeq_cohort(cohort, page.Size)
```

outputs a data frame of all available sample-level log2 miRSeq expression values of the cohort TGCT.

The functions of *dn_miRNA.R* and *dn_mRNA.R* are constructed analogously.

2.2.1 Reshape

The log2 expression values, variable `expression_log2`, corresponding to a patient barcode and a mir is the most important information provided by the sample-level log2 miRSeq and mRNASeq data. This consideration resulted in the development of reshaping algorithms for these data types.

The file *reshape.R* contains the functions *reshape.*{*miRSeq*, *mRNASeq*} used to reshape sample-level log2 miRSeq and mRNASeq expression values of the sample type TP. I.e., first use the functions from *dn_miRNA.R* or *dn_mRNA.R* to download the input parameter `data` of *reshape.miRSeq* or *reshape.mRNASeq* before reshaping.

The function *reshape.miRSeq* with the input parameter `data` return a nxp-matrix $M = (m_{ij})_{i=1,..,n,j=1,...,p}$, n = number of patients and p = number of mirs, rownames of the matrix correspond to the patients, colnames correspond to the mirs, so the following applies

$$m_{ij} = (expression\_log2)_{ij} \text{ where}$$
$$data\$tcga\_participant\_barcode == barcode[i] \quad \wedge \quad data\$mir == mir[j]$$

Primarily this service is intended to reshape all sample-level log2 miRSeq expression values of one cohort.

Specify the `cohort` to query, e.g. TGCT, download all available sample-level log2 miRSeq expression values of this cohort, as described in 2.2. Reshape the data frame `tgct.miRSeq` as follows

```
> data = tgct.miRSeq
> tgct.miRSeq_reshaped = reshape.miRSeq (data,
sample_type = "TP")
```

A small subset of the `tgct.miRSeq_reshaped`:

```
> tgct.miRSeq_reshaped
         row.names        hsa-let-7a-2-3p      hsa-let-7a-3p
1        TCGA-2G-AAFZ     2.4124445            4.033826
2        TCGA-2G-AAFY     3.0375198            4.378292
3        TCGA-2G-AAFV     2.8927868            3.794833
```

reshape.mRNASeq is built in the exact same manner as *reshape.miRSeq*.

2.3 Selected columns from the MAF generated by MutSig

The file *dn_mutation.R* provides the download of selected columns from the MAF generated by MutSig. It consists of the functions *dn_*{*mutation.Exp*, *dn_mutation_cohort*}. All data frames of selected columns from the MAF generated by MutSig mainly provide information about the variables `Hugo_Symbol (= gene)`, `Variant_Type`, `Variant_Classification`, `Protein_Change`, `SwissProt_entry_Id`, `Tumor_Sample_Barcode (= patient barcode)`, `tool` and `cohort`. All variables can be seen on the FireBrowse website[6].

As for other data types the first function is intended to download rather small data whereas the second function is a service for downloading all available selected columns from the MAF generated by MutSig for one cohort.

[6] `http://firebrowse.org/api-docs/#!/Metadata/MAFColNames`

Using *dn_mutation.Exp* you can query data for single or multiple input arguments, namely `genes, cohorts, barcodes` and `tools`. You have to specify at least one `gene` or `barcode` or `cohort`.

```
> tcga_participant_barcode = "TCGA-AG-A002"
   # a TCGA patient barcode from READ
> cohort = "READ"
> gene = c("A1BG", "A1CF", "A2M")
> page.Size = 250
> sort_by = "gene"
> tool = "MutSig2CV"
> obj = dn_mutation.Exp(tcga_participant_barcode, cohort, gene,
page.Size, sort_by, tool)
```

The `obj` is a data frame with 3 observations of 8 variables.

```
> obj
Hugo_Symbol Variant_Type Variant_Classification    Protein_Change
1      A1CF          SNP                  Intron
2      A2M           SNP       Nonsense_Mutation          p.E840*
3      A2M           SNP       Missense_Mutation          p.P969S

 SwissProt_entry_Id          Tumor_Sample_Barcode       tool  cohort
1        A1CF_HUMAN  TCGA-AG-A002-01A-01W-A00K-09  MutSig2CV    READ
2         2MG_HUMAN  TCGA-AG-A002-01A-01W-A00K-09  MutSig2CV    READ
3        A2MG_HUMAN  TCGA-AG-A002-01A-01W-A00K-09  MutSig2CV    READ
```

If you like to download all available data for one cohort, use *dn_mutation_cohort* with input parameters `cohort` and `page.Size`. You can also download this data applying *dn_mutation.Exp* just specifying a `cohort`. But the download by means of *dn_mutation_cohort* is much faster as applying *dn_mutation.Exp*.

An example how to use *dn_mutation_cohort*:

```
> cohort = "READ"
> page.Size = 500
> read.mutation = dn_mutation_cohort(cohort, page.Size)
```

2.4 Significantly mutated genes, as scored by MutSig

The file *dn_mutation_SMG.R* provides significantly mutated genes, as scored by MutSig. It is structured in the same way as other services. The file consists of the functions *dn_{mutation.SMG.Exp, mutation.SMG_cohort}*. *dn_mutation.SMG.Exp* with input arguments `gene, rank, cohort, page.Size` and `sort_by` returns data for rather small specific querries, the service *dn_mutation.SMG_cohort* with input parameters `cohort` and `page.Size` returns all available data from the MAF generated by MutSig for one cohort.

Examples how to apply the functions:

```
> gene = c("A1BG", "A1CF")
> rank = ""
> cohort = "ACC"
> page.Size = 100
> sort_by = "gene"
> obj = dn_mutation.SMG.Exp(gene, rank, cohort, page.Size, sort_by)
```

The `obj` is a data frame of 3 observations of 23 variables. An extract of `obj`:

```
> obj[,c("gene", "cohort", "longname", "p", "sample_type")]
gene   cohort                          longname          p sample_type
1 A1BG    ACC          alpha-1-B glycoprotein  0.3485129          TP
2 A1CF    ACC  APOBEC1 complementation factor  1.0000000          TP
3  A2M    ACC           alpha-2-macroglobulin  1.0000000          TP
```

The data frame of all significantly mutated genes for one cancer, e.g. PCPG, can be downloaded as follows:

```
> cohort = "PCPG"
> page.Size = 1000
> pcpg.mutation.SMG = dn_mutation.SMG_cohort(cohort, page.Size)
```

3 Example: Sample-level log2 miRSeq expression values in conjunction with the "glmnet" R Package

In this section we will analyse the sample-level log2 miRSeq expression values of the cancer adrenocortical carcinoma (ACC) and show which `mirs (genes)` are selected with respect to the vital status (alive, dead) of the patients by applying the logistic regression.

First download ACC clinical data to get the TCGA patient barcodes of ACC and the related vital status.

```
> cohort = "ACC"
> acc.clinical = dn_clinical_one(cohort)
```

At the present ACC comprises 92 patients in total.
Afterwords extract the information about the vital status of the patient barcodes from `acc.clinical`:

```
> acc.label = acc.clinical[,c("tcga_participant_barcode",
"vital_status")]
```

The column "vital_status" of `acc.label`, namely the labels "alive" and "dead" as characters, has to be converted to factors:

```
> acc.label$vital_status = factor(acc.label$vital_status)
```

Next, we download all available sample-level log2 miRSeq expression values of "ACC" as described in section 2.2.:

```
> page.Size = 1000
> acc.miRSeq = dn_miRSeq_cohort(cohort, page.Size)
```

Then reshape `acc.miRSeq` as described in 2.2.1:

```
> acc.miRSeq_reshaped = reshape.miRSeq(acc.miRSeq,sample_type="TP")
```

Many matrix elements of `acc.miRSeq_reshaped` are NA's, these entries correspond to the mirs whose log2 miRSeq expression values are "None", see `acc.miRSeq`. As "glmnet" cannot deal with NA's, for purposes of demonstration we remove all columns containing NA's from `acc.miRSeq_reshaped` and store the resulting matrix as `X`

```
> nc = ncol(acc.miRSeq_reshaped)
> na.ind = sapply(1:nc,function(i){any(is.na(acc.miRSeq_reshaped[,i]
  ))})
> X = acc.miRSeq_reshaped[,!na.ind]
```

Note that the sample-level log2 miRSeq expression values are available just for 80 patients. Therefore, to perform the analysis it is necessary to select these patients from the clinical data, more precisely from `acc.label`, and to ensure the same order of the barcodes from `X` and the filtered barcodes from `acc.label`, see STEP1 and STEP2 in the code below:

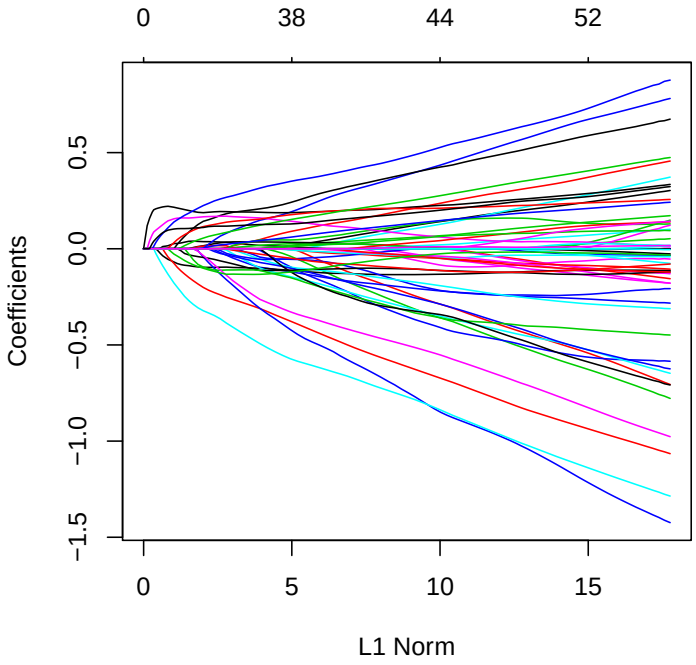
```
> barcode = rownames(X)
> y = acc.label$vital_status
> names(y) = acc.label$tcga_participant_barcode
> idx = match(barcode, names(y)) # STEP1
> y = y[idx,drop=FALSE] # STEP2
```

Now the preparation for "glmnet" is done and the actual analysis can be performed. Apply "glmnet" to X and y and plot the result to illustrate the change of the coefficients, see the figure on the next page.

```
> out = cv.glmnet(X, y, family="binomial", intercept=FALSE)
> plot(out)
```

The next steps deliver significant mirs.

```
> c = as.numeric(coef(out2)[-1])
> names(c) = colnames(X)
> idx = order(abs(c), decreasing=TRUE)
> c = c[idx,drop=FALSE]
> idx = abs(c) > 0
> data.frame(coef=c[idx])
                            coef
hsa-mir-874-3p   -1.584823e-01
hsa-mir-7-1-3p    1.032142e-01
hsa-mir-181c-3p  -6.623627e-02
hsa-mir-424-3p    6.393274e-02
hsa-mir-509-3-5p -2.417862e-02
hsa-mir-196b-5p   2.359241e-02
hsa-mir-92b-3p    6.039862e-05
```



According to the result, the selected mirs above affect the vital status most, especially the mir "hsa-mir-874-3p". In spite of the relatively rough analysis the result is still suitable as you can find information about some of the selected mirs in conjunction with ACC, especially the mir "hsa-mir-7-1-3p" or at least the mir-7 microRNA precursor[7], seems to play an important role[8].

4 Conclusion

This technical report introduced the R package *EasyTCGA* allowing easy batch downloading of particular data types of TCGA from FireBrowse mentioned in the introduction. We presented all functions of the package and its applications. The package is quite easy to use, just some input parameters have to be specified, whereby almost all relevant input parameters are listed or linked in the documentation or are provided by the package itself.

One drawback of the package is that the download may take a while, since the FireBrowse service itself is not designed for batch downloading. Especially the download of all available data for a cohort can take some time. But this is primarily due to the Firebrowse API. Depending on the API being used, the service may return NULL although the data for your query is available.So whenever it happens, we recommend you to double check on FireBrowse website for data availability[9].

[7] https://en.wikipedia.org/wiki/Mir-7_microRNA_precursor
[8] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4742203/
https://www.google.com/patents/EP2657348A1?cl=en
[9] http://firebrowse.org/

Acknowledgement

This work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project C1.